Doc No.: D3077R0 DRAFT 1
Author: Andrew Tomazos <andrewtomazos@gmail.com>
Date: 2023-12-30
Project: Programming Language - C++ (WG21)
Audience: Evolution Working Group

Proposal of static_cast shorthand: <T>(x)

Summary

We propose making the static_cast keyword optional in a static cast.

Background

It is a well-known best practice to favor static_casts over C-style casts, because static_casts provide more safety and are less error-prone:

That is, to prefer:

```
static_cast<T>(x)
```

to:

```
(T)x
```

The syntax of static_cast is quite verbose considering how common it is. There is on average over one static_cast per C++ source file. (Over a million hits in ACTCD19)

static_cast is obviously more verbose than the C-style cast.

Proposal

To address this we propose to make the `static_cast` keyword optional:

```
<T>(x)  // equivalent to static_cast<T>(x)
```

Examples

Example 1

```
// today
temp = static_cast<uint8_t>(temp << 4) | static_cast<uint8_t>(*p - '0');

// proposed
temp = <uint8_t>(temp << 4) | <uint8_t>(*p - '0');
```

Example 2

```
// today
code_flags[static_cast<u16>(addr + opcode->size)] |= CODE_CHECK_INT;

// proposed
code_flags[<u16>(addr + opcode->size)] |= CODE_CHECK_INT;
```

Example 3

```
// today
log->Printf("Process::%s (arg = %p, pid = %" PRIu64
                        ") about to exit with internal state %s...",
                        __FUNCTION__, static_cast<void *>(this), GetID(),
                        StateAsCString(internal_state));

// proposed
log->Printf("Process::%s (arg = %p, pid = %" PRIu64
                        ") about to exit with internal state %s...",
                        __FUNCTION__, <void *>(this), GetID(),
                        StateAsCString(internal_state));
```

Wording

postfix-expression:
  static_cast$_{opt}$ < type-id > (expression)

Implementation

An expression today cannot start with a < token, therefore if an expression does start with a < token, it unambiguously begins the proposed <T>(x) expression. Parsing and semantic analysis of the remaining expression following the < token proceeds identically to todays static_cast expression. Implementation is therefore trivial.